LISTING OF CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Currently Amended) A method for evaluating a plurality of candidate index sets for a workload of database statements in a database system, the method comprising:

forming an index superset from a union of a current index set and a proposed index set;

deriving a candidate index set from the index superset, the derived candidate index set being incorporated into the plurality of candidate index sets;

analyzing collected database statistics based on the derived candidate index set;

repeatedly deriving a candidate index set and analyzing collected statistics based on the proposed index set;

terminating the repeated execution when at least one candidate index solution is found that adheres to user-imposed constraints and no further indexes can be removed from said candidate index solution without degrading performance of the workload and without disabling an integrity constraint, wherein the integrity constraint describes a condition about the database that must always be true or must always be false; and

presenting the analyzed collected statistics.

2. (Previously Presented) The method of claim 1, further comprising:

deriving current index statistics for the workload responsive to the current index set, the presented collected statistics comprising the derived current index statistics.

3. (Canceled)

4. (Canceled)

5. (Previously Presented) The method of claim 56, wherein analyzing the collected baseline statistics comprises disabling current indexes.

6. (Previously Presented) The method of claim 1, wherein analyzing statistics for a statement comprises generating at least one statistic based on an execution plan created by an optimizer.

7. (Previously Presented) The method of claim 6, wherein the execution plan is based on available access paths.

8. (Previously Presented) The method of claim 6, wherein the execution plan is based on statistics for at least one schema object accessed by the statement.

9. (Original) The method of claim 8 wherein the at least one schema object is a table.

10. (Original) The method of claim 8 wherein the at least one schema object is an index.

11. (Previously Presented) The method of claim 6, further comprising:

for a table accessed by a statement under evaluation, using the execution plan to identify at least one index that would be used to retrieve data from the table upon an execution of the statement.

12. (Previously Presented) The method of claim 6, wherein the optimizer generates a cost of the execution plan.

13. (Original) The method of claim 12, wherein the cost of the execution plan is derived from a resource use needed to execute the statement according to the execution plan.

14. (Original) The method of claim 13, wherein the resource use includes CPU execution time.

15. (Original) The method of claim 13, wherein the resource use includes input/output access.

16. (Previously Presented) The method of claim 6, wherein the collected database statistics comprise the number of executions of the statement.

17. (Previously Presented) The method of claim 6, wherein the collected database statistics comprise a user-defined importance of the statement.

18. (Previously Presented) The method of claim 6, wherein the collected database statistics comprise an index usage.

19. (Canceled)

20. (Original) The method of claim 1, wherein the statements are SQL statements.

21. (Original) The method of claim 1, wherein the workload is reduced into unique statements.

22. (Original) The method of claim 1, wherein deriving a candidate index set is responsive to a predetermined maximum number of allowed indexes.

23. (Original) The method of claim 1, wherein deriving a candidate index set is responsive to available storage space.

24. (Original) The method of claim 1, wherein the proposed index set is provided by a user.

25. (Original) The method of claim 1, wherein the proposed index set is provided by an expert system.

26. (Original) The method of claim 1, wherein an execution plan is created without creating indexes which are not in the current index set.

27. (Currently Amended) A system for evaluating a plurality of candidate index sets for a workload in a database system, the workload derived from a plurality of statements, the system comprising:

a workload evaluator, including a processor, wherein the workload evaluator ~~which~~ evaluates each statement within the workload using collected database statistics;

an index solution evaluator which, responsive to the workload evaluator, evaluates each index in a candidate index set with respect to the workload, the candidate index solution being one of the plurality of candidate index sets, each candidate index set derived from an index superset formed by the union of a current index set and a proposed index set;

a solution/rollup evaluator which, responsive to the index solution evaluator, evaluates the candidate index solution; and

a solution refiner which, responsive to the solution/rollup evaluator, generates at least one new candidate index solution, the at least one new candidate index solution being incorporated into the plurality of candidate index sets, wherein the solution refiner further generates at least one new candidate index solution by eliminating at least one index on a small table under evaluation, wherein the at least one index does not enforce an integrity constraint, wherein the integrity constraint describes a condition about the database that must always be true or must always be false.

28. (Original) The system of claim 27, wherein the solution refiner generates at least one new candidate index solution by eliminating at least one index within the candidate index solution that does not adhere to user-imposed constraints.

29. (Original) The system of claim 28, wherein the constraint is a user-defined constraint.

30. (Original) The system of claim 28, wherein the constraint is a memory-usage constraint.

31. (Canceled).

32. (Previously Presented) The system of claim 27, wherein the workload evaluator evaluates an execution plan created by an optimizer, the execution plan comprising, for each statement of the workload, an execution plan which represents a series of steps for executing the statement, the workload evaluator further generating and recording statistics based on the evaluation of the execution plan.

33. (Original) The system of claim 32, wherein each execution plan is created based on available access paths.

34. (Original) The system of claim 32, wherein each execution plan is created based on statistics for at least one schema object accessed by the statement.

35. (Original) The system of claim 34 wherein the at least one schema object is a table.

36. (Original) The system of claim 34 wherein the at least one schema object is an index.

37. (Original) The system of claim 32, wherein the workload evaluator, for a table accessed by a statement under evaluation, identifies at least one index which would be used to retrieve data from the table upon an execution of the statement.

38. (Original) The system of claim 32, wherein the workload evaluator determines a cost of the execution plan.

39. (Original) The system of claim 38, wherein the cost of the execution plan is derived from a resource use needed to execute the statement according to the execution plan.

40. (Original) The system of claim 39, wherein the resource use includes CPU execution time.

41. (Original) The system of claim 39, wherein the resource use includes input/output access.

42. (Original) The system of claim 32, wherein the statistics include the number of executions of the statement.

43. (Original) The system of claim 32, wherein the statistics include a user-defined importance of the statement.

44. (Original) The system of claim 32, wherein the statistics include an index usage.

45. (Original) The system of claim 32, wherein the statistics include a cost of the execution plan.

46. (Original) The system of claim 27, wherein the statements are SQL statements.

47. (Original) The system of claim 27, wherein the workload is reduced into unique statements.

48. (Original) The system of claim 27, wherein the solution refiner is responsive to a predetermined maximum number of allowed indexes.

49. (Original) The system of claim 27, wherein the solution refiner is responsive to available storage space.

50. (Original) The system of claim 27, wherein the proposed index set is provided by a user.

51. (Original) The system of claim 27, wherein the proposed index set is provided by an expert system.

52. (Original) The system of claim 27, wherein an execution plan is created without creating indexes which are not in the current index set.

53. (Currently Amended) A computer program product for evaluating a plurality of candidate index sets for a workload of database statements in a database system, the computer program product comprising a computer usable medium having computer readable code thereon, including program code which:

forms an index superset from a union of a current index set and a proposed index set;

repeatedly derives a candidate index set from the index superset, the derived candidate index set being incorporated into the plurality of candidate index sets, generates statistics based on the proposed index set, and analyzes collected database statistics based on the derived candidate index set;

terminates the repeated execution when at least one candidate index solution is found that adheres to user-imposed constraints and no further indexes can be removed from said candidate index solution without degrading performance of the workload and without disabling an integrity constraint, wherein the integrity constraint describes a condition about the database that must always be true or must always be false; and

presents the analyzed statistics.

54. (Canceled)

55. (Previously Presented) The method of claim 1, wherein the statistics include index volatility.

56. (Previously Presented) The method of claim 1, further comprising:

generating baseline statistics for each statement in the workload, wherein generating statistics is additionally based on the baseline statistics.

57. (Previously Presented) The system of claim 32, wherein the statistics include index volatility.

58. (Previously Presented) The computer program product of claim 53, further comprising code which:

derives current index statistics for the workload responsive to the current index set, the presented analyzed statistics comprising the derived current index statistics.

59. (Canceled)

60. (Canceled)

61. (Previously Presented) The computer program product of claim 53, wherein the statistics include index volatility.

62. (Previously Presented) The computer program product of claim 53, further comprising code which:

derives baseline statistics for each statement in the workload, wherein deriving statistics is additionally based on the baseline statistics.

63. (Previously Presented) The computer program product of claim 62, wherein deriving the baseline statistics comprises disabling current indexes.

64. (Previously Presented) The computer program product of claim 53, wherein statistics for a statement are derived by generating at least one statistic based on an execution plan created by an optimizer.

65. (Previously Presented) The computer program product of claim 64, wherein the execution plan is based on available access paths.

66. (Previously Presented) The computer program product of claim 64, wherein the execution plan is based on statistics for at least one schema object accessed by the statement.

67. (Previously Presented) The computer program product of claim 66 wherein the at least one schema object is a table.

68. (Previously Presented) The computer program product of claim 66 wherein the at least one schema object is an index.

69. (Previously Presented) The computer program product of claim 64, further comprising code which:

for a table accessed by a statement under evaluation, uses the execution plan to identify at least one index that would be used to retrieve data from the table upon an execution of the statement.

70. (Previously Presented) The computer program product of claim 64, wherein the optimizer generates a cost of the execution plan.

71. (Previously Presented) The computer program product of claim 70, wherein the cost of the execution plan is derived from a resource use needed to execute the statement according to the execution plan.

72. (Previously Presented) The computer program product of claim 71, wherein the resource use comprises CPU execution time.

73. (Previously Presented) The computer program product of claim 71, wherein the resource use comprises input/output access.

74. (Previously Presented) The computer program product of claim 64, wherein the statistics comprise the number of executions of the statement.

75. (Previously Presented) The computer program product of claim 64, wherein the statistics comprise a user-defined importance of the statement.

76. (Previously Presented) The computer program product of claim 64, wherein the statistics comprise an index usage.

77. (Previously Presented) The computer program product of claim 53, wherein the statements are SQL statements.

78. (Previously Presented) The computer program product of claim 53, wherein the workload is reduced into unique statements.

79. (Previously Presented) The computer program product of claim 53, wherein a candidate index set is derived responsive to a predetermined maximum number of allowed indexes.

80. (Previously Presented) The computer program product of claim 53, wherein deriving a candidate index set is responsive to available storage space.

81. (Previously Presented) The computer program product of claim 53, wherein the proposed index set is provided by a user.

82. (Previously Presented) The computer program product of claim 53, wherein the proposed index set is provided by an expert system.

83. (Previously Presented) The computer program product of claim 53, wherein an execution plan is created without creating indexes which are not in the current index set.

84. (Canceled)

85. (Canceled)

86. (Canceled)

87. (Canceled)

88. (Canceled)

89. (Canceled)

90. (Canceled)

91. (Canceled)

92. (Canceled)

93. (Canceled)

94. (Canceled)

95. (Canceled)

96. (Canceled)

97. (Canceled)

98. (Canceled)

99. (Canceled)

100. (Canceled)

101. (Canceled)

102. (Canceled)

103. (Canceled)

104. (Canceled)

105. (Canceled)

106. (Canceled)

107. (Canceled)

108. (Canceled)

109. (Canceled)

110. (Currently Amended) A system for evaluating a plurality of candidate index sets for a workload of database statements in a database system, comprising:

means for forming an index superset from a union of a current index set and a proposed index set, wherein the means include a processor;

means for deriving a candidate index set from the index superset, the derived candidate index set being incorporated into the plurality of candidate index sets;

means for analyzing collected database statistics based on the derived candidate index set;

means for repeatedly deriving a candidate index set and generating statistics based on the proposed index set;

means for terminating the repeated execution when at least one candidate index solution is found that adheres to user-imposed constraints and no further indexes can be removed from said candidate index solution without degrading performance of the workload and without disabling an integrity constraint, wherein the integrity constraint describes a condition about the database that must always be true or must always be false; and

means for presenting the analyzed statistics.

111. (Previously Presented) The system of claim 110, further comprising:

means for generating current index statistics for the workload responsive to the current index set, the presented generated statistics comprising the generated current index statistics.

112. (Canceled)

113. (Canceled)

114. (Previously Presented) The system of claim 110, wherein the statistics include index volatility.

115. (Previously Presented) The system of claim 110, further comprising:

means for generating baseline statistics for each statement in the workload, wherein statistics are generated based additionally on the baseline statistics.

116. (Previously Presented) The system of claim 115, wherein analyzing the baseline statistics comprises disabling current indexes.

117. (Previously Presented) The system of claim 110, wherein analyzing statistics for a statement comprises generating at least one statistic based on an execution plan created by an optimizer.

118. (Previously Presented) The system of claim 117, wherein the execution plan is based on available access paths.

119. (Previously Presented) The system of claim 117, wherein the execution plan is based on statistics for at least one schema object accessed by the statement.

120. (Previously Presented) The system of claim 119 wherein the at least one schema object is a table.

121. (Previously Presented) The system of claim 119 wherein the at least one schema object is an index.

122. (Previously Presented) The system of claim 117, further comprising:

means for using, for a table accessed by a statement under evaluation, the execution plan to identify at least one index that would be used to retrieve data from the table upon an execution of the statement.

123. (Previously Presented) The system of claim 117, wherein the optimizer generates a cost of the execution plan.

124. (Previously Presented) The system of claim 123, wherein the cost of the execution plan is derived from a resource use needed to execute the statement according to the execution plan.

125. (Previously Presented) The system of claim 124, wherein the resource use comprises CPU execution time.

126. (Previously Presented) The system of claim 124, wherein the resource use comprises input/output access.

127. (Previously Presented) The system of claim 117, wherein the statistics comprise the number of executions of the statement.

128. (Previously Presented) The system of claim 117, wherein the statistics comprise a user-defined importance of the statement.

129. (Previously Presented) The system of claim 117, wherein the statistics comprise an index usage.

130. (Previously Presented) The system of claim 110, wherein the statements are SQL statements.

131. (Previously Presented) The system of claim 110, wherein the workload is reduced into unique statements.

132. (Previously Presented) The system of claim 110, wherein deriving a candidate index set is responsive to a predetermined maximum number of allowed indexes.

133. (Previously Presented) The system of claim 110, wherein deriving a candidate

index set is responsive to available storage space.

134. (Previously Presented) The system of claim 110, wherein the proposed index set is provided by a user.

135. (Previously Presented) The system of claim 110, wherein the proposed index set is provided by an expert system.

136. (Previously Presented) The system of claim 110, wherein an execution plan is created without creating indexes which are not in the current index set.

137. (New) The method of claim 1 wherein terminating comprises:

terminating the repeated execution when at least one candidate index solution is found that adheres to user-imposed constraints and no further indexes can be removed from said candidate index solution without degrading performance of the workload and without disabling an integrity constraint, wherein the integrity constraint describes a condition about the database that must always be true or must always be false, and wherein any removed index that does not enforce an integrity constraint is disabled.

138. (New) The system of claim 27 wherein a solution refiner comprises:

a solution refiner which, responsive to the solution/rollup evaluator, generates at least one new candidate index solution, the at least one new candidate index solution being incorporated into the plurality of candidate index sets, wherein the solution refiner further generates at least one new candidate index solution by eliminating at least one index on a small table under evaluation, wherein the at least one index does not enforce an integrity constraint, wherein the integrity constraint describes a condition about the database that must always be true or must always be false, and wherein any eliminated index that does not enforce an integrity constraint is disabled.

139. (New) The computer program product of claim 53 wherein program code which terminates comprises:

terminates the repeated execution when at least one candidate index solution is found that adheres to user-imposed constraints and no further indexes can be removed from said candidate index solution without degrading performance of the workload and without disabling an integrity constraint, wherein the integrity constraint describes a condition about the database that must always be true or must always be false, and wherein any removed index that does not enforce an integrity constraint is disabled.

140. (New) The system of claim 110 wherein means for terminating comprises:

means for terminating the repeated execution when at least one candidate index solution is found that adheres to user-imposed constraints and no further indexes can be removed from said candidate index solution without degrading performance of the workload and without disabling an integrity constraint, wherein the integrity constraint describes a condition about the database that must always be true or must always be false, and wherein any removed index that does not enforce an integrity constraint is disabled.